

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-03/msg02162.html>

- *From:* Balbir Singh <balbir@xxxxxxxxxx>
 - *Date:* Tue, 7 Mar 2006 22:56:59 +0530
-

On Mon, Feb 27, 2006 at 08:17:47PM +1100, Nick Piggin wrote:

Shailabh Nagar wrote:

schedstats-sysctl.patch

Add sysctl option for controlling schedstats collection dynamically. Delay accounting leverages schedstats for cpu delay statistics.

I'd sort of rather not tie this in with schedstats if possible. Schedstats adds a reasonable amount of cache footprint and branches in hot paths. Most of schedstats stuff is something that hardly anyone will use.

Sure you can share common code though...

This patch refines scheduler statistics collection and display to three levels, tasks, runqueue and scheddomains. CONFIG_SCHEDSTATS now requires a boot time option in the form of schedstats or schedstats= to display scheduler statistics

They can all be enabled together to get complete statistics by passing "all" as a boot time option. schedstat_inc has been split into schedstat_rq_inc and schedstat_sd_inc, each of which checks if rq and sd statistics gathering is enabled or not. schedstat_add has been changed to schedstat_sd_add, it checks if sd statistics gathering is on prior to gathering the statistics. Similar changes have been made for task schedstats gathering.

The output of /proc/schedstat and /proc/<pid>/schedstat and /proc/<pid>/task/*/schedstat has been modified to print a gentle message suggesting that statistics gathering is off. Also a header "statistics for cpuXXX" has been added (for each cpu) to the /proc/schedstat output.

This patch is motivated by comments for sharing code with

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

CONFIG_SCHEDSTATS but not incurring the complete overhead of the entire CONFIG_SCHEDSTATS code.

Testing

=====

a) booted with schedstats (schedstats=all)

```
cat /proc/schedstat
version 13
timestamp 4294922919
```

statistics for cpu0

```
cpu0 10 10 132 142 240 97775 24181 48251 45935 5664 2376 73594
domain0 00000003 24464 24242 162 224 62 4 0 24242 149 148 0 1 1 0 0 148 24290 24117 64 173 109 0 0
24117 0 0 0 0 0 0 0 0 0 4392 835 0
```

statistics for cpu1

```
cpu1 3 3 180 14504 387 50735 6520 15430 11035 4195 2376 44215
domain0 00000003 25870 25203 107 695 588 3 0 25203 198 198 0 0 0 0 198 6608 6428 91 184 93 0 0 6428
0 0 0 0 0 0 0 0 2316 307 0
```

```
cat /proc/1/schedstat
506 34 1102
```

b) booted with schedstats=tasks

```
cat /proc/schedstat
version 13
timestamp 4294937241
runqueue and scheddomain stats are not enabled
```

```
cat /proc/1/schedstat
505 58 1097
```

c) booted with schedstats=rq

```
cat /proc/schedstat
version 13
timestamp 4294913832
```

statistics for cpu0

```
cpu0 14 14 56 102 260 96332 18867 47278 45064 3556397949 2216 77465
scheddomain stats are not enabled
```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

statistics for cpu1

cpu1 3 3 12134 12138 333 42878 4224 12874 8779 1714457722 2071 38654
scheddomain stats are not enabled

cat /proc/1/schedstat
tasks schedstats is not enabled

d) booted with schedstats=sd

cat /proc/schedstat
version 13
timestamp 4294936220

statistics for cpu0

runqueue stats are not enabled
domain0 00000003 38048 37802 140 248 108 0 0 37802 151 149 0 3 3 0 0 149 27574 27417 59 158 99 0 0
27417 0 0 0 0 0 0 0 0 4168 827 0

statistics for cpu1

runqueue stats are not enabled
domain0 00000003 39094 38441 119 682 563 3 0 38441 199 196 0 5 5 0 0 196 9167 8970 107 203 96 0 0
8970 0 0 0 0 0 0 0 0 2159 330 0

cat /proc/1/schedstat
tasks schedstats is not enabled

Alternatives considered

=====

The other alternative that was considered was that instead of changing the format of /proc/schedstat and /proc/<pid>*/schedstat, we could print zeros for all levels for which statistics is not collected. But zeros could be treated as valid values, so this solution was not implemented.

Limitations

=====

The effectiveness of this patch is limited to run-time statistics collection. The run-time overhead is proportional to the level of statistics enabled. The space consumed is the same as before.

This patch was created against 2.6.16-rc5

Signed-off-by: Balbir Singh <balbir@xxxxxxxxxx>

```

Documentation/kernel-parameters.txt | 11 ++
fs/proc/base.c | 4
include/linux/sched.h | 23 +++++
kernel/sched.c | 195 ++++++-----
4 files changed, 175 insertions(+), 58 deletions(-)

```

```

diff -puN kernel/sched.c~schedstats_refinement kernel/sched.c
--- linux-2.6.16-rc5/kernel/sched.c~schedstats_refinement 2006-03-07 16:14:59.000000000 +0530
+++ linux-2.6.16-rc5-balbir/kernel/sched.c 2006-03-07 20:43:46.000000000 +0530
@@ -386,7 +386,28 @@ static inline void task_rq_unlock(runque
* bump this up when changing the output format or the meaning of an existing
* format, so that tools can adapt (or abort)
*/
-#define SCHEDSTAT_VERSION 12
+#define SCHEDSTAT_VERSION 13
+
+int schedstats_on __read_mostly;
+
+/*
+ * Parse the schedstats options passed at boot time
+ */
+static int __init schedstats_setup_enable(char *str)
+{
+ if (!str || !strcmp(str, "") || !strcmp(str, "=all"))
+ schedstats_on = SCHEDSTATS_ALL;
+ else if (!strcmp(str, "=tasks"))
+ schedstats_on = SCHEDSTATS_TASKS;
+ else if (!strcmp(str, "=sd"))
+ schedstats_on = SCHEDSTATS_SD;
+ else if (!strcmp(str, "=rq"))
+ schedstats_on = SCHEDSTATS_RQ;
+
+ return 1;
+}
+
+__setup("schedstats", schedstats_setup_enable);

static int show_schedstat(struct seq_file *seq, void *v)
{
@@ -394,26 +415,44 @@ static int show_schedstat(struct seq_fil
seq_printf(seq, "version %d\n", SCHEDSTAT_VERSION);
seq_printf(seq, "timestamp %lu\n", jiffies);
+
+ if (!schedstats_rq_on() && !schedstats_sd_on()) {
+ seq_printf(seq, "runqueue and scheddomain stats are not "
+ "enabled\n");
+ return 0;
+ }

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```

+
for_each_online_cpu(cpu) {
runqueue_t *rq = cpu_rq(cpu);
#ifdef CONFIG_SMP
struct sched_domain *sd;
int dcnt = 0;
#endif
+ seq_printf(seq, "\nstatistics for cpu%d\n", cpu);

- /* runqueue-specific stats */
- seq_printf(seq,
- "cpu%d %lu %lu %lu %lu %lu %lu %lu %lu %lu %lu %lu",
- cpu, rq->yld_both_empty,
- rq->yld_act_empty, rq->yld_exp_empty, rq->yld_cnt,
- rq->sched_switch, rq->sched_cnt, rq->sched_goidle,
- rq->ttwu_cnt, rq->ttwu_local,
- rq->rq_sched_info.cpu_time,
- rq->rq_sched_info.run_delay, rq->rq_sched_info.pcnt);
+ if (schedstats_rq_on()) {
+ /* runqueue-specific stats */
+ seq_printf(seq,
+ "cpu%d %lu %lu %lu %lu %lu %lu %lu %lu %lu %lu %lu "
+ "%lu",
+ cpu, rq->yld_both_empty,
+ rq->yld_act_empty, rq->yld_exp_empty, rq->yld_cnt,
+ rq->sched_switch, rq->sched_cnt, rq->sched_goidle,
+ rq->ttwu_cnt, rq->ttwu_local,
+ rq->rq_sched_info.cpu_time,
+ rq->rq_sched_info.run_delay, rq->rq_sched_info.pcnt);

- seq_printf(seq, "\n");
+ seq_printf(seq, "\n");
+ } else
+ seq_printf(seq, "runqueue stats are not enabled\n");

#ifdef CONFIG_SMP
+
+ if (!schedstats_sd_on()) {
+ seq_printf(seq, "scheddomain stats are not enabled\n");
+ continue;
+ }
+
+ /* domain-specific stats */
preempt_disable();
for_each_domain(cpu, sd) {
@@ -472,11 +511,30 @@ struct file_operations proc_schedstat_op
.release = single_release,
};

-# define schedstat_inc(rq, field) do { (rq)->field++; } while (0)
-# define schedstat_add(rq, field, amt) do { (rq)->field += (amt); } while (0)

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```

+# define schedstat_sd_inc(sd, field) \
+do { \
+ if (unlikely(schedstats_sd_on())) \
+ (sd)->field++; \
+} while (0)
+
+# define schedstat_sd_add(sd, field, amt) \
+do { \
+ if (unlikely(schedstats_sd_on())) \
+ (sd)->field += (amt); \
+} while (0)
+
+# define schedstat_rq_inc(rq, field) \
+do { \
+ if (unlikely(schedstats_rq_on())) \
+ (rq)->field++; \
+} while (0)
+
+
+/* !CONFIG_SCHEDSTATS */
+# define schedstat_inc(rq, field) do { } while (0)
+# define schedstat_add(rq, field, amt) do { } while (0)
+
+# define schedstat_sd_inc(rq, field) do { } while (0)
+# define schedstat_sd_add(rq, field, amt) do { } while (0)
+# define schedstat_rq_inc(rq, field) do { } while (0)
+
+
+#endif

/*
@@ -515,6 +573,15 @@ static inline void sched_info_dequeued(t
t->sched_info.last_queued = 0;
}

+static void rq_sched_info_arrive(struct runqueue *rq, unsigned long diff)
+{
+ if (!schedstats_rq_on() || !rq)
+ return;
+
+ rq->rq_sched_info.run_delay += diff;
+ rq->rq_sched_info.pcnt++;
+}
+
+/*
+ * Called when a task finally hits the cpu. We can now calculate how
+ * long it was waiting to run. We also note when it began so that we
@@ -523,20 +590,23 @@ static inline void sched_info_dequeued(t
static void sched_info_arrive(task_t *t)
{
unsigned long now = jiffies, diff = 0;
- struct runqueue *rq = task_rq(t);

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```

+ if (!schedstats_tasks_on() && !schedstats_rq_on())
+ return;
+
+ /*
+ * diff is required in case schedstats is on for tasks or rq
+ */
if (t->sched_info.last_queued)
diff = now - t->sched_info.last_queued;
sched_info_dequeued(t);
- t->sched_info.run_delay += diff;
- t->sched_info.last_arrival = now;
- t->sched_info.pcnt++;
-
- if (!rq)
- return;

- rq->rq_sched_info.run_delay += diff;
- rq->rq_sched_info.pcnt++;
+ if (schedstats_tasks_on()) {
+ t->sched_info.run_delay += diff;
+ t->sched_info.last_arrival = now;
+ t->sched_info.pcnt++;
+ }
+ rq_sched_info_arrive(task_rq(t), diff);
}

/*
@@ -556,23 +626,32 @@ static void sched_info_arrive(task_t *t)
*/
static inline void sched_info_queued(task_t *t)
{
+ if (!schedstats_tasks_on() && !schedstats_rq_on())
+ return;
+
if (!t->sched_info.last_queued)
t->sched_info.last_queued = jiffies;
}

+static inline void rq_sched_info_depart(struct runqueue *rq, unsigned long diff)
+{
+ if (!schedstats_rq_on() || !rq)
+ return;
+
+ rq->rq_sched_info.cpu_time += diff;
+}
+
+/*
+ * Called when a process ceases being the active-running process, either
+ * voluntarily or involuntarily. Now we can calculate how long we ran.
+ */
static inline void sched_info_depart(task_t *t)

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```
{
- struct runqueue *rq = task_rq(t);
unsigned long diff = jiffies - t->sched_info.last_arrival;

- t->sched_info.cpu_time += diff;
-
- if (rq)
- rq->rq_sched_info.cpu_time += diff;
+ if (schedstats_tasks_on())
+ t->sched_info.cpu_time += diff;
+ rq_sched_info_depart(task_rq(t), diff);
}

/*
@@ -1190,15 +1269,15 @@ static int try_to_wake_up(task_t *p, uns

new_cpu = cpu;

- schedstat_inc(rq, ttwu_cnt);
+ schedstat_rq_inc(rq, ttwu_cnt);
if (cpu == this_cpu) {
- schedstat_inc(rq, ttwu_local);
+ schedstat_rq_inc(rq, ttwu_local);
goto out_set_cpu;
}

for_each_domain(this_cpu, sd) {
if (cpu_isset(cpu, sd->span)) {
- schedstat_inc(sd, ttwu_wake_remote);
+ schedstat_sd_inc(sd, ttwu_wake_remote);
this_sd = sd;
break;
}
@@ -1239,7 +1318,7 @@ static int try_to_wake_up(task_t *p, uns
* p is cache cold in this domain, and
* there is no bad imbalance.
*/
- schedstat_inc(this_sd, ttwu_move_affine);
+ schedstat_sd_inc(this_sd, ttwu_move_affine);
goto out_set_cpu;
}
}
@@ -1250,7 +1329,7 @@ static int try_to_wake_up(task_t *p, uns
*/
if (this_sd->flags & SD_WAKE_BALANCE) {
if (imbalance*this_load <= 100*load) {
- schedstat_inc(this_sd, ttwu_move_balance);
+ schedstat_sd_inc(this_sd, ttwu_move_balance);
goto out_set_cpu;
}
}
}
```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

@@ -1894,7 +1973,7 @@ skip_queue:

```
#ifdef CONFIG_SCHEDSTATS
```

```
if (task_hot(tmp, busiest->timestamp_last_tick, sd))
```

```
- schedstat_inc(sd, lb_hot_gained[idle]);
```

```
+ schedstat_sd_inc(sd, lb_hot_gained[idle]);
```

```
#endif
```

```
pull_task(busiest, array, tmp, this_rq, dst_array, this_cpu);
```

@@ -1913,7 +1992,7 @@ out:

```
* so we can safely collect pull_task() stats here rather than
```

```
* inside pull_task().
```

```
*/
```

```
- schedstat_add(sd, lb_gained[idle], pulled);
```

```
+ schedstat_sd_add(sd, lb_gained[idle], pulled);
```

```
if (all_pinned)
```

```
*all_pinned = pinned;
```

@@ -2109,23 +2188,23 @@ static int load_balance(int this_cpu, ru

```
if (idle != NOT_IDLE && sd->flags & SD_SHARE_CPUPOWER)
```

```
sd_idle = 1;
```

```
- schedstat_inc(sd, lb_cnt[idle]);
```

```
+ schedstat_sd_inc(sd, lb_cnt[idle]);
```

```
group = find_busiest_group(sd, this_cpu, &imbalance, idle, &sd_idle);
```

```
if (!group) {
```

```
- schedstat_inc(sd, lb_nobusy[idle]);
```

```
+ schedstat_sd_inc(sd, lb_nobusy[idle]);
```

```
goto out_balanced;
```

```
}
```

```
busiest = find_busiest_queue(group, idle);
```

```
if (!busiest) {
```

```
- schedstat_inc(sd, lb_nobusyq[idle]);
```

```
+ schedstat_sd_inc(sd, lb_nobusyq[idle]);
```

```
goto out_balanced;
```

```
}
```

```
BUG_ON(busiest == this_rq);
```

```
- schedstat_add(sd, lb_imbalance[idle], imbalance);
```

```
+ schedstat_sd_add(sd, lb_imbalance[idle], imbalance);
```

```
nr_moved = 0;
```

```
if (busiest->nr_running > 1) {
```

@@ -2146,7 +2225,7 @@ static int load_balance(int this_cpu, ru

```
}
```

```
if (!nr_moved) {
```

```
- schedstat_inc(sd, lb_failed[idle]);
```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```

+ schedstat_sd_inc(sd, lb_failed[idle]);
sd->nr_balance_failed++;

if (unlikely(sd->nr_balance_failed > sd->cache_nice_tries+2)) {
@@ -2199,7 +2278,7 @@ static int load_balance(int this_cpu, ru
return nr_moved;

out_balanced:
- schedstat_inc(sd, lb_balanced[idle]);
+ schedstat_sd_inc(sd, lb_balanced[idle]);

sd->nr_balance_failed = 0;

@@ -2233,22 +2312,22 @@ static int load_balance_newidle(int this
if (sd->flags & SD_SHARE_CPUPOWER)
sd_idle = 1;

- schedstat_inc(sd, lb_cnt[NEWLY_IDLE]);
+ schedstat_sd_inc(sd, lb_cnt[NEWLY_IDLE]);
group = find_busiest_group(sd, this_cpu, &imbalance, NEWLY_IDLE, &sd_idle);
if (!group) {
- schedstat_inc(sd, lb_nobusyq[NEWLY_IDLE]);
+ schedstat_sd_inc(sd, lb_nobusyq[NEWLY_IDLE]);
goto out_balanced;
}

busiest = find_busiest_queue(group, NEWLY_IDLE);
if (!busiest) {
- schedstat_inc(sd, lb_nobusyq[NEWLY_IDLE]);
+ schedstat_sd_inc(sd, lb_nobusyq[NEWLY_IDLE]);
goto out_balanced;
}

BUG_ON(busiest == this_rq);

- schedstat_add(sd, lb_imbalance[NEWLY_IDLE], imbalance);
+ schedstat_sd_add(sd, lb_imbalance[NEWLY_IDLE], imbalance);

nr_moved = 0;
if (busiest->nr_running > 1) {
@@ -2260,7 +2339,7 @@ static int load_balance_newidle(int this
}

if (!nr_moved) {
- schedstat_inc(sd, lb_failed[NEWLY_IDLE]);
+ schedstat_sd_inc(sd, lb_failed[NEWLY_IDLE]);
if (!sd_idle && sd->flags & SD_SHARE_CPUPOWER)
return -1;
} else
@@ -2269,7 +2348,7 @@ static int load_balance_newidle(int this
return nr_moved;

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```
out_balanced:
- schedstat_inc(sd, lb_balanced[NEWLY_IDLE]);
+ schedstat_sd_inc(sd, lb_balanced[NEWLY_IDLE]);
if (!sd_idle && sd->flags & SD_SHARE_CPUPOWER)
return -1;
sd->nr_balance_failed = 0;
@@ -2333,12 +2412,12 @@ static void active_load_balance(runqueue
if (unlikely(sd == NULL))
goto out;

- schedstat_inc(sd, alb_cnt);
+ schedstat_sd_inc(sd, alb_cnt);

if (move_tasks(target_rq, target_cpu, busiest_rq, 1, sd, SCHED_IDLE, NULL))
- schedstat_inc(sd, alb_pushed);
+ schedstat_sd_inc(sd, alb_pushed);
else
- schedstat_inc(sd, alb_failed);
+ schedstat_sd_inc(sd, alb_failed);
out:
spin_unlock(&target_rq->lock);
}
@@ -2906,7 +2985,7 @@ need_resched_nonpreemptible:
dump_stack();
}

- schedstat_inc(rq, sched_cnt);
+ schedstat_rq_inc(rq, sched_cnt);
now = sched_clock();
if (likely((long long)(now - prev->timestamp) < NS_MAX_SLEEP_AVG)) {
run_time = now - prev->timestamp;
@@ -2974,7 +3053,7 @@ go_idle:
/*
* Switch the active and expired arrays.
*/
- schedstat_inc(rq, sched_switch);
+ schedstat_rq_inc(rq, sched_switch);
rq->active = rq->expired;
rq->expired = array;
array = rq->active;
@@ -3007,7 +3086,7 @@ go_idle:
next->activated = 0;
switch_tasks:
if (next == rq->idle)
- schedstat_inc(rq, sched_goidle);
+ schedstat_rq_inc(rq, sched_goidle);
prefetch(next);
prefetch_stack(next);
clear_tsk_need_resched(prev);
@@ -3979,7 +4058,7 @@ asmlinkage long sys_sched_yield(void)
```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```
prio_array_t *array = current->array;
prio_array_t *target = rq->expired;

- schedstat_inc(rq, yld_cnt);
+ schedstat_rq_inc(rq, yld_cnt);
/*
* We implement yielding by moving the task into the expired
* queue.
@@ -3991,11 +4070,11 @@ asmlinkage long sys_sched_yield(void)
target = rq->active;

if (array->nr_active == 1) {
- schedstat_inc(rq, yld_act_empty);
+ schedstat_rq_inc(rq, yld_act_empty);
if (!rq->expired->nr_active)
- schedstat_inc(rq, yld_both_empty);
+ schedstat_rq_inc(rq, yld_both_empty);
} else if (!rq->expired->nr_active)
- schedstat_inc(rq, yld_exp_empty);
+ schedstat_rq_inc(rq, yld_exp_empty);

if (array != target) {
dequeue_task(current, array);
diff -puN include/linux/sched.h~schedstats_refinement include/linux/sched.h
--- linux-2.6.16-rc5/include/linux/sched.h~schedstats_refinement 2006-03-07 16:14:59.000000000 +0530
+++ linux-2.6.16-rc5-balbir/include/linux/sched.h 2006-03-07 20:46:10.000000000 +0530
@@ -525,6 +525,29 @@ struct backing_dev_info;
struct reclaim_state;

#ifdef CONFIG_SCHEDSTATS
+
+#define SCHEDSTATS_TASKS 0x1
+#define SCHEDSTATS_RQ 0x2
+#define SCHEDSTATS_SD 0x4
+#define SCHEDSTATS_ALL (SCHEDSTATS_TASKS | SCHEDSTATS_RQ | SCHEDSTATS_SD)
+
+extern int schedstats_on;
+
+static inline int schedstats_tasks_on(void)
+{
+ return schedstats_on & SCHEDSTATS_TASKS;
+}
+
+static inline int schedstats_rq_on(void)
+{
+ return schedstats_on & SCHEDSTATS_RQ;
+}
+
+static inline int schedstats_sd_on(void)
+{
+ return schedstats_on & SCHEDSTATS_SD;

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

```
+}
+
struct sched_info {
/* cumulative counters */
unsigned long cpu_time, /* time spent on the cpu */
diff -puN Documentation/kernel-parameters.txt~schedstats_refinement
Documentation/kernel-parameters.txt
--- linux-2.6.16-rc5/Documentation/kernel-parameters.txt~schedstats_refinement 2006-03-07
16:14:59.000000000 +0530
+++ linux-2.6.16-rc5-balbir/Documentation/kernel-parameters.txt 2006-03-07 20:48:44.000000000 +0530
@@ -1333,6 +1333,17 @@ running once the system is up.
sc1200wdt= [HW,WDT] SC1200 WDT (watchdog) driver
Format: <io>[,<timeout>[,<isapnp>]]
```

```
+ schedstats [KNL]
+ Enable all schedstats if CONFIG_SCHEDSTATS is defined
+ same as the schedstats=all
+
+ schedstats= [KNL]
+ Format: {"all", "tasks", "sd", "rq"}
+ all --- turns on the complete schedstats
+ rq --- turns on schedstats only for runqueue
+ sd --- turns on schedstats only for scheddomains
+ tasks -- turns on schedstats only for tasks
+
scsi_debug_*= [SCSI]
See drivers/scsi/scsi_debug.c.
```

```
diff -puN fs/proc/base.c~schedstats_refinement fs/proc/base.c
--- linux-2.6.16-rc5/fs/proc/base.c~schedstats_refinement 2006-03-07 16:14:59.000000000 +0530
+++ linux-2.6.16-rc5-balbir/fs/proc/base.c 2006-03-07 16:17:39.000000000 +0530
@@ -72,6 +72,7 @@
#include <linux/cpuset.h>
#include <linux/audit.h>
#include <linux/poll.h>
+#include <linux/sched.h>
#include "internal.h"

/*
@@ -504,6 +505,9 @@ static int proc_pid_wchan(struct task_st
*/
static int proc_pid_schedstat(struct task_struct *task, char *buffer)
{
+ if (!schedstats_tasks_on())
+ return sprintf(buffer, "tasks schedstats is not enabled\n");
+
return sprintf(buffer, "%lu %lu %lu\n",
task->sched_info.cpu_time,
task->sched_info.run_delay,
-
-

```

schedstats refinement (was Re: [Lse-tech] Re: [Patch 2/7] Add sysctl for schedstats)

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>